

Description:

The simplest example of using MOAB is to load a mesh and iterate over its entities. The first step to using MOAB is to instantiate the library, calling the MBCore constructor. Lists of entity handles can be stored efficiently using MOAB's MBRange class. After retrieving the list of 3D entities in the mesh, that list is iterated using C++ STL-like functions in MBRange.

The [makefile.in](#) file in the MOAB source is processed by MOAB's autotools-based build, producing a makefile like the following:

```
#
# The following few lines get the library install location for MOAB
# and puts it in MOAB_LIB_DIR; if you just want
# a simple application makefile, just comment out the next 3 lines and
# substitute your own value for $(MOAB_LIB_DIR), or set an environment
# variable pointing to the right place.
exec_prefix = ${prefix}
prefix = /home/tautges/MOABclean
libdir = ${exec_prefix}/lib
MOAB_LIB_DIR = $(libdir)

include $(MOAB_LIB_DIR)/moab.make

CPPFLAGS = ${MOAB_INCLUDES}
CXXFLAGS = -g

default: all

all: GetEntities

GetEntities: GetEntities.o
    $(CXX) -o $@ $< $(MOAB_LIBS_LINK)

GetEntities.o : GetEntities.cpp
    $(CXX) -c $(MOAB_INCLUDES) $<
```

This makefile shows three key requirements for building MOAB-based applications:

1. Use the make 'include' command to include MOAB's moab.make file, by default installed at the same time and location of the MOAB libraries. In the above makefile, this location is `$(MOAB_LIB_DIR)`. Applications can input that location directly, as an environment or make variable, or, as in the `makefile.in` file, using autotools-generated variables.
2. Use the make variable `$(MOAB_INCLUDES)` as compile options for application files depending on MOAB functions. This make variable, defined in `moab.make`, contains compiler options telling the location of MOAB include files, and other CPP variable definitions required to use them.
3. Use the make variable `$(MOAB_LIBS_LINK)` as a link option for applications; this pulls in the MOAB libraries and any other 3rd-party libraries it depends on.

For a more detailed explanation of these concepts, see the MOAB v4.0 User's Guide.

[Code](#)
[makefile.in](#)

Description:

Test file

To run:

GetEntities brick_cubit10.2.cub

Output:

```
Found 1331 0-dimensional entities:
Found d=0 entity 1.
Found d=0 entity 2.
...
Found d=0 entity 1331.
Found 1200 1-dimensional entities:
Found d=1 entity 10.
Found d=1 entity 11.
...
Found d=1 entity 1379.
Found 600 2-dimensional entities:
Found d=2 entity 100.
Found d=2 entity 101.
...
Found d=2 entity 699.
Found 1000 3-dimensional entities:
Found d=3 entity 1.
Found d=3 entity 2.
```